WirePlumber 0.5

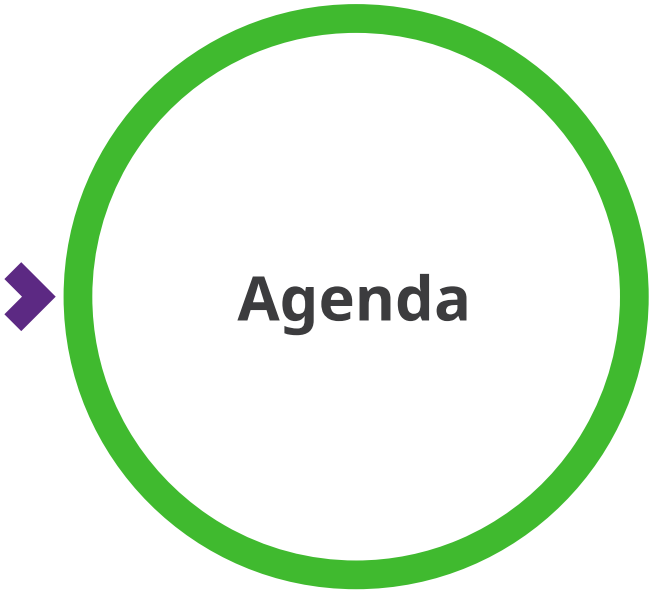
A smart way to handle Audio Filters

Julian Bouzas
Senior Software Engineer

April 2024 – Seattle

Hi, I am Julian

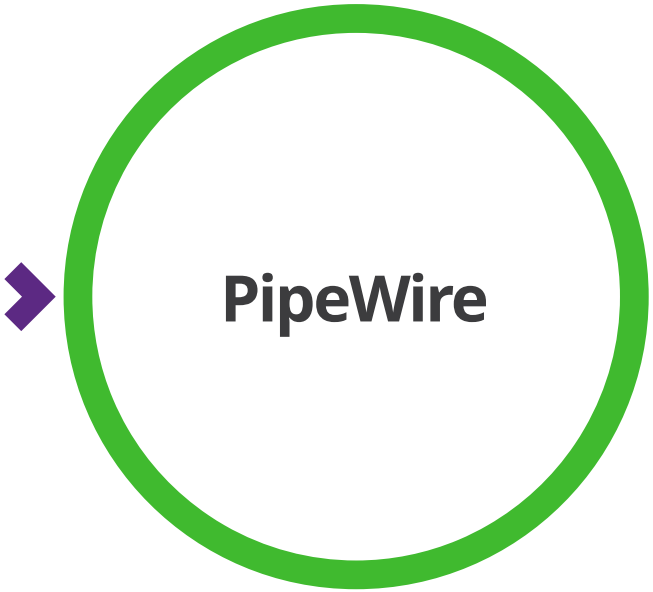
- From Spain
- Multimedia Team at Collabora since 2019
- GStreamer, PipeWire and WirePlumber developer
- julian.bouzas@collabora.com



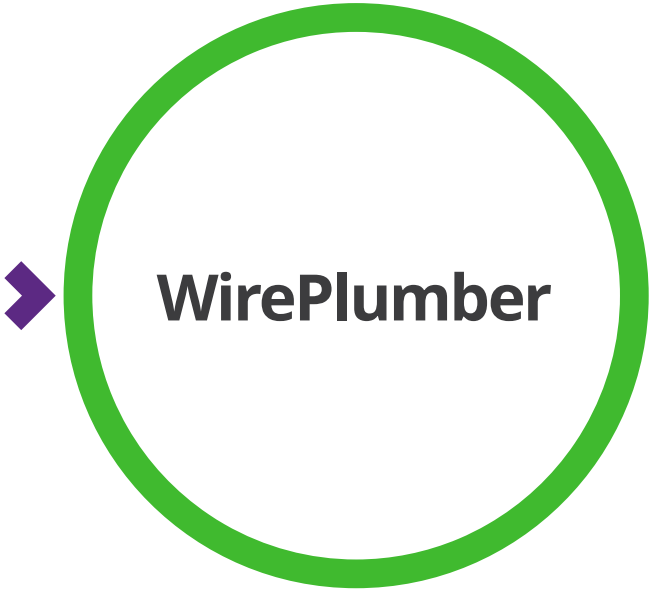
- Intro to PipeWire and WirePlumber
- WirePlumber 0.5 Release
- PipeWire Audio Filters
- Smart Filters Policy in WirePlumber
- Smart Filters Example for Bluetooth
- Demo

COLLABORA

Intro to PipeWire and WirePlumber

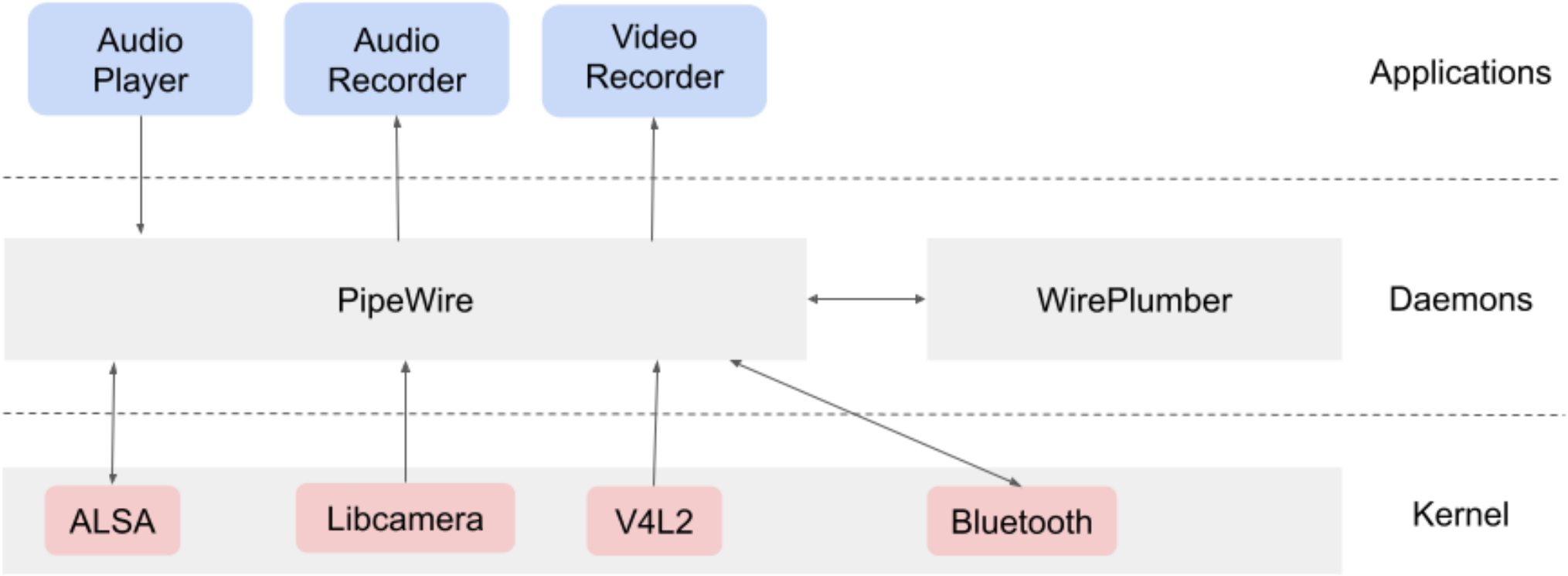


- Next Generation Audio and Video Server to handle multimedia devices on Linux
- Daemon, like PulseAudio but for Video as well
- Known for its exceptional performance and low latency
- Built with Security in mind
- External Session Manager
  - Adaptable for any use-case (both Embedded and Desktop)



- Default Session Manager for PipeWire
- Sponsored by Collabora
- Configures and handles PipeWire objects:
  - Configures device profiles and routes
  - Defines access permissions between objects
  - Remembers all the above user preferences across reboots
  - Link objects to form the processing graph (Policy)
- Extensible, modular and extremely configurable
  - Lua scripts implement the management functionality

Multimedia Stack



COLLABORA

WirePlumber 0.5 Release

WirePlumber 0.5

- New major release
- 2 years of development since 0.4 was released
- Clean API to hopefully last for a long time (more like 1.0)
- Plenty of new features

WirePlumber 0.5 Features

- Event Dispatcher
  - Addresses many event race conditions in a more robust and elegant way
- JSON Configuration
  - More consistency with PipeWire
- Settings
  - Change behavior at runtime
- Smart Filters Policy
  - Automatic handling of audio filters

Collabora blog post

- https://www.collabora.com/news-and-blog/blog/2024/02/19/whats-the-latest-with-wireplumber

Status

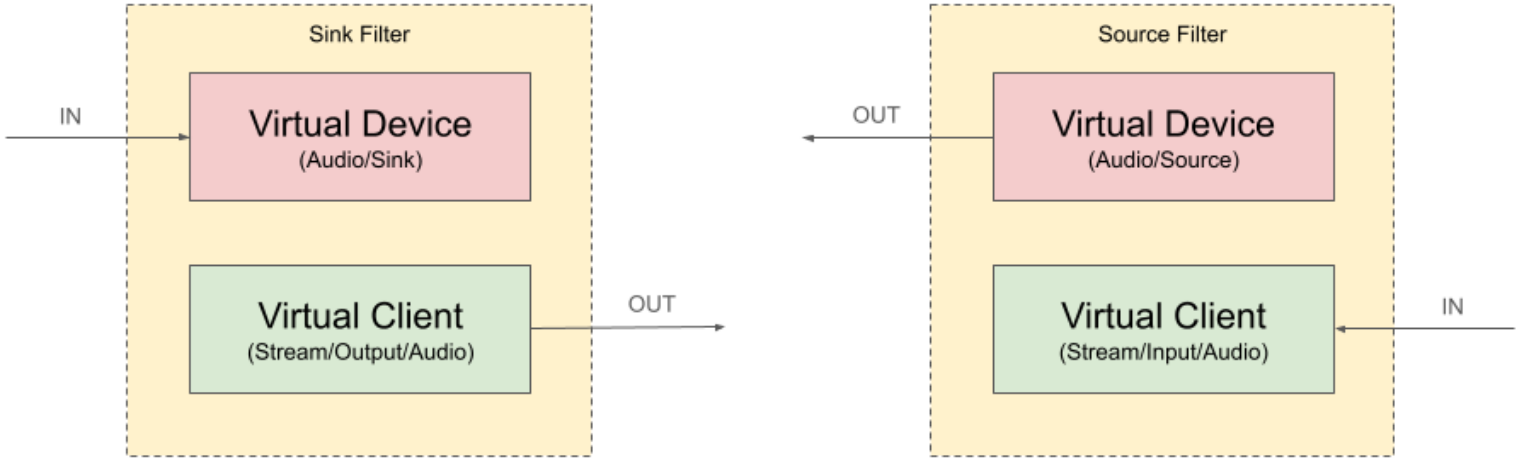
- Latest stable release 0.5.1 (Bug fix)
- MIT License
- Hosted on: https://gitlab.freedesktop.org/pipewire/wireplumber
- Included by default in major Linux distributions:
  - Fedora
  - Debian (Unstable)
  - Arch Linux

COLLABORA

PipeWire Audio Filters

Audio Filters in PipeWire

- Processing unit that transforms audio signals
- Defined as a pair of nodes by PipeWire

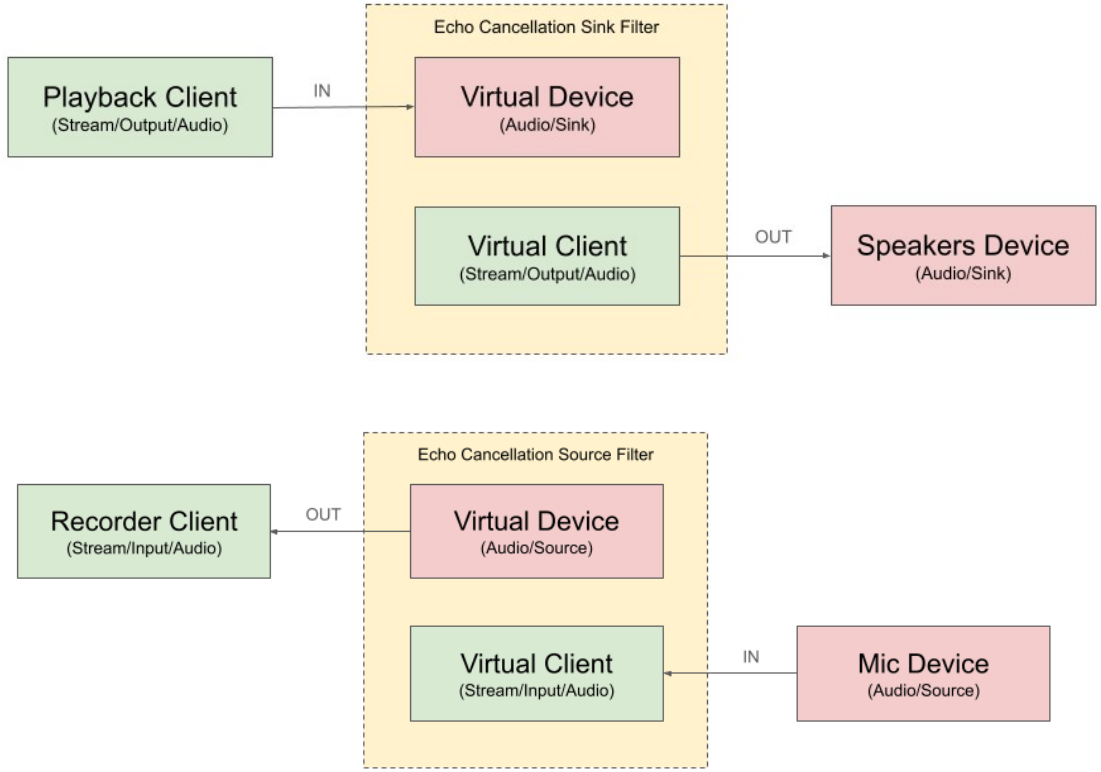


Example of Audio Filters in PipeWire

- Loopback
- Filter Chain (chains LADSPA, LV2 and builtin filters)
  - Equalizers
  - Compressors
  - Limiters
- Noise Reduction
- Echo Cancellation

Echo Cancellation

- 2 filters (4 nodes)
  - Sink filter
  - Source filter
- Source filter cancels audio monitored from sink filter



How are filters handled?

- Manual user intervention with WirePlumber 0.4
- Cumbersome if users want to use only a specific filter for a particular device
  - Example: Only use echo-cancel-sink filter for the speakers device, and the echo-cancel-source filter for the built-in microphone device.
- Automatic with WirePlumber 0.5 Smart Filters Policy

Smart Filters Policy in WirePlumber

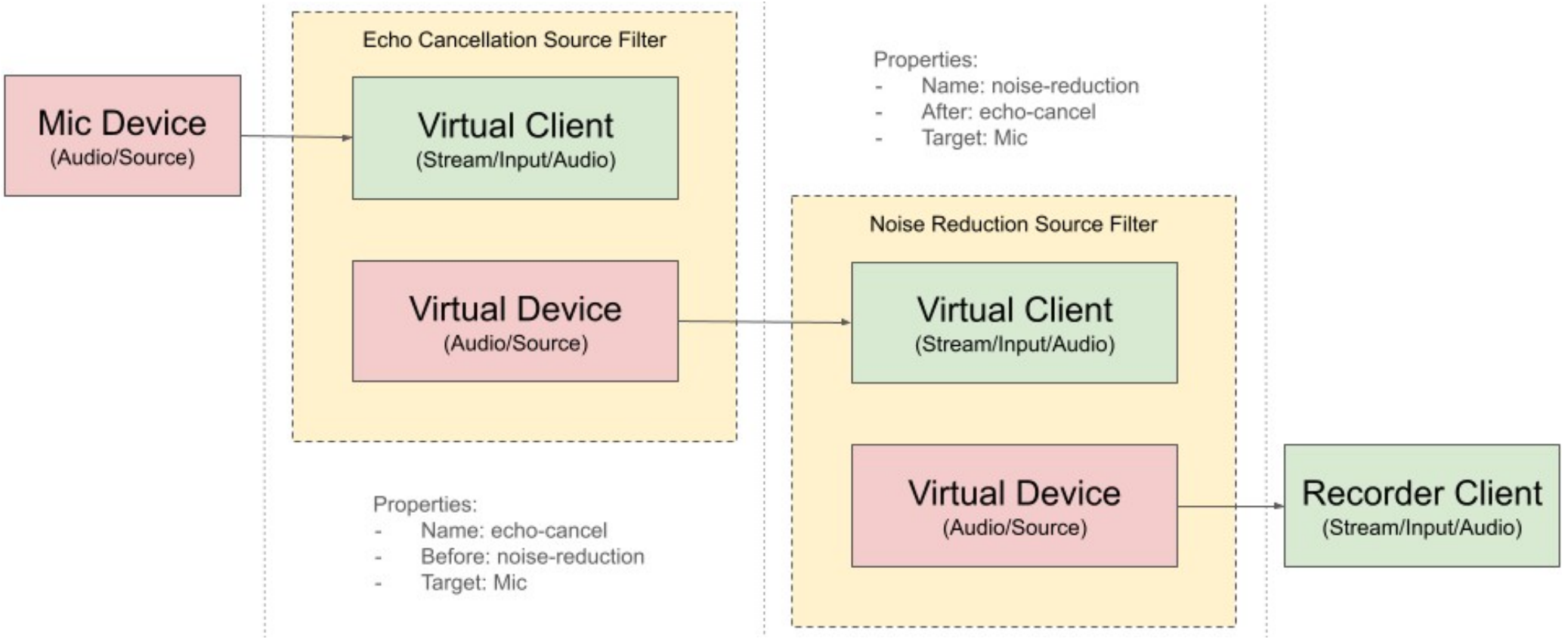
Smart Filter Properties

- **filter.smart:** Whether to mark the filter as smart or not
- **filter.smart.name:** Unique filter identifier
- **filter.smart.before:** List of filters that should be used before this one
- **filter.smart.after:** List of filters that should be used after this one
- **filter.smart.target:** Properties matching the final target node in the chain

Smart Filters Configuration

- Can be permanent if set in the PipeWire JSON configuration
- Can be changed at runtime easily using the metadata tool
  - $ pw-metadata -n filters ...

```
context.modules = [
  {
    name = libpipewire-module-loopback
    args = {
      node.name = virtual-sink
      node.description = "Virtual Sink"
      capture.props = {
        media.class = Audio/Sink
        audio.position = [ FL FR ]
        filter.smart = true
        filter.smart.name = loopback-sink
        filter.smart.target = {
          node.name = "Speakers"
        }
      }
      playback.props = {
        media.class = Stream/Output/Audio
        audio.position = [ FL FR ]
        node.passive = true
        node.dont-remix = true
      }
    }
  }
]
```

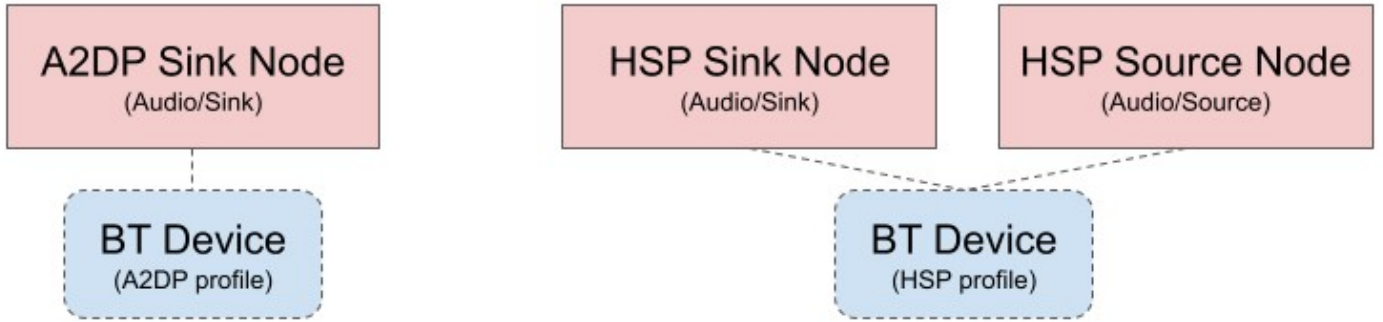

Mic Device
(Audio/Source)
Echo Cancellation Source Filter
Virtual Client
(Stream/Input/Audio)
Virtual Device
(Audio/Source)
Properties:
- Name: echo-cancel
- Before: noise-reduction
- Target: Mic
Properties:
- Name: noise-reduction
- After: echo-cancel
- Target: Mic
Noise Reduction Source Filter
Virtual Client
(Stream/Input/Audio)
Virtual Device
(Audio/Source)
Recorder Client
(Stream/Input/Audio)

COLLABORA

Smart Filters Example for Bluetooth

BT profile auto-switch example

- Most Bluetooth devices support 2 profiles:
  - A2DP: High quality audio (playback)
  - HSP: low latency profile for communication (capture and playback)



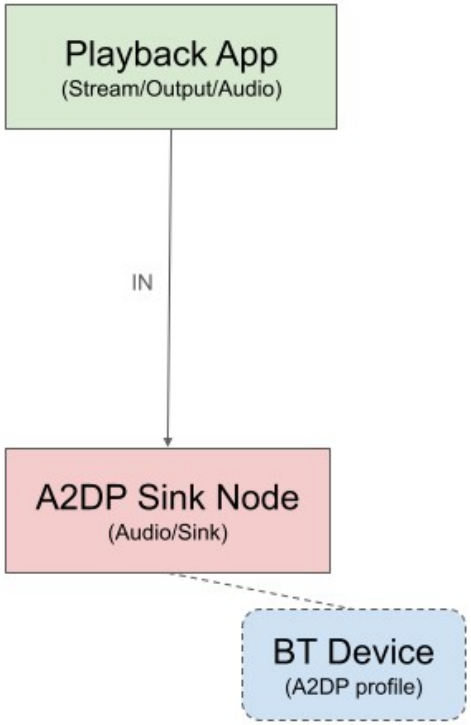
Handling of BT profiles

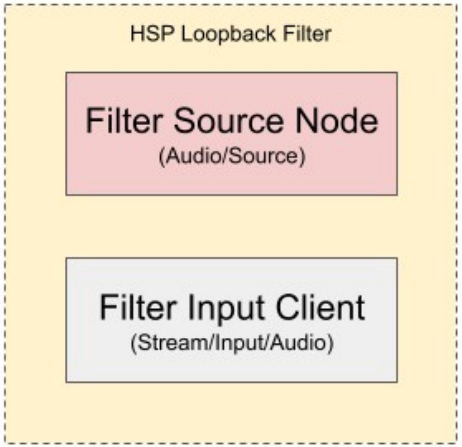
- Since there is no BT source node when A2DP profile is selected, applications can’t know if the BT device supports HSP profile
  - 0.4 series: auto-switch to HFP profile if client with specific names (Zoom, Firefox, etc...) want to capture audio when the BT device is configured as the default device.
- Ideally, we want to use A2DP profile only when there is no application that wants to capture audio from the BT device
  - 0.5 series: can do that with loopback filter and smart filter policy

Handling of BT profiles (continued)

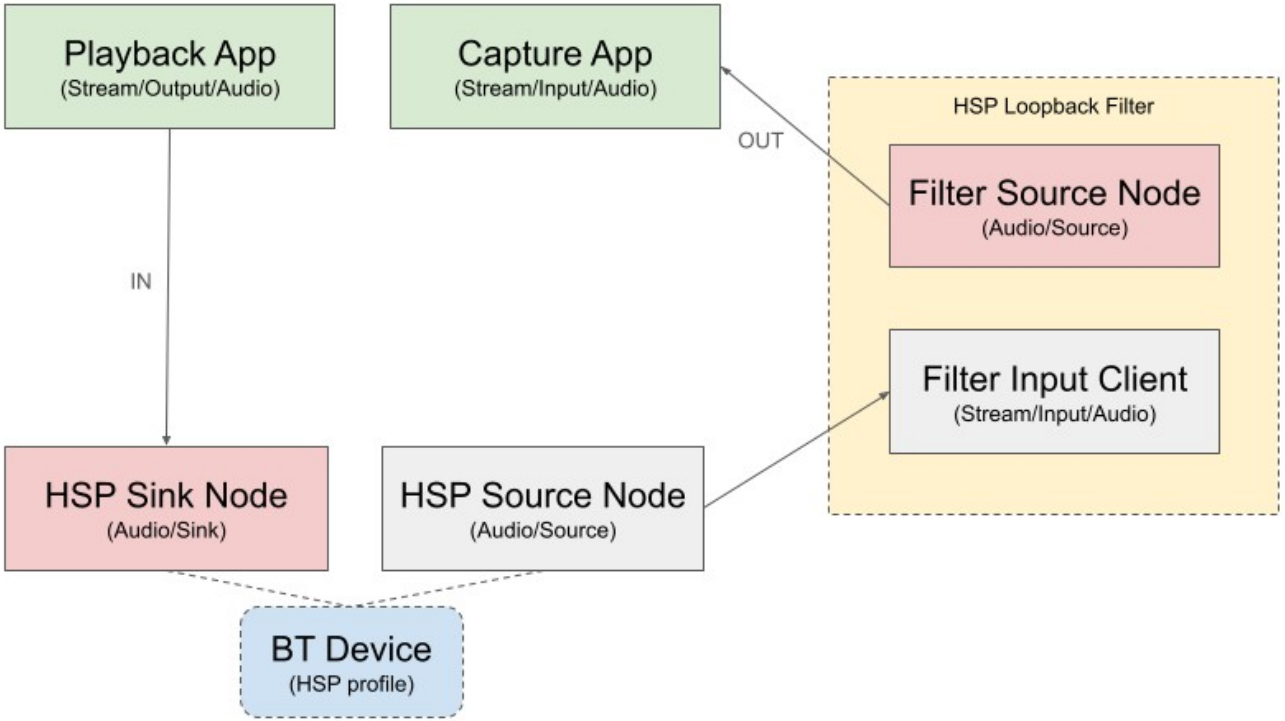
- WirePlumber creates 1 loopback source filter per connected BT device that supports HSP profile
- The target of the loopback source filter is set to the HSP source node if any
- The actual HSP source node is always hidden from applications
- HSP/A2DP profile auto-switch happens when a client node starts/stops capturing audio from the loopback source filter node


A2DP
Profile


Playback App
(Stream/Output/Audio)
IN
A2DP Sink Node
(Audio/Sink)
BT Device
(A2DP profile)


HSP Loopback Filter
Filter Source Node
(Audio/Source)
Filter Input Client
(Stream/Input/Audio)

HSP Profile


Playback App
(Stream/Output/Audio)
Capture App
(Stream/Input/Audio)
OUT
HSP Loopback Filter
Filter Source Node
(Audio/Source)
IN
Filter Input Client
(Stream/Input/Audio)
HSP Sink Node
(Audio/Sink)
HSP Source Node
(Audio/Source)
BT Device
(HSP profile)

BT profile auto-switch advantages

- Applications don’t need to worry about what BT profile is set on any device
- Works for multiple connected BT devices, even if they are not configured as the default device
- Any application can capture from any BT device without user intervention
  - No need to rely on hardcoded application names (Zoom, Firefox, etc...)

COLLABORA

Demo

Thank you!